Message Queue Formats

How does the message queue work?

Sep 14, 1988

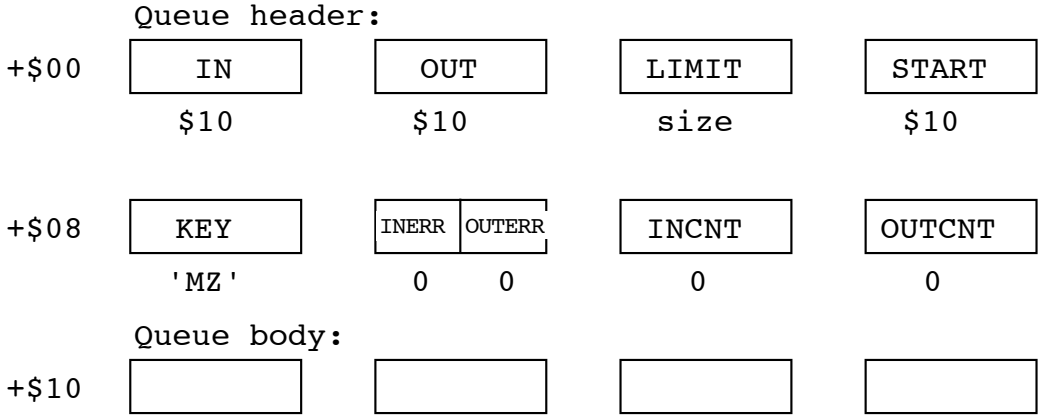
This note describes the format of the shared memory Message Queue used for communication with another processor on the VMEbus. It is implemented as a simple one-way queue. Messages are placed in the queue by the VME System computer. The other processor removes messages from the queue and interprets the command accordingly.

Initialization

The VME System computer initializes co-processor message queues at reset time. One of the standard system tables—table #15—contains pointers to the message queue for each co-processor. This queue pointer table, indexed by co-processor number, has 8-byte entries of the following format:

The message queue pointer is followed by the total queue size.

Each co-processor queue has the following format:



The values beneath the words in the queue header are the initialized values. The other processor, when it recognizes the presence of the queue, examines the `KEY` field. If it has the value '`MZ`' it changes it to '`MQ`' to signal to the VME system cpu that it has "seen" the message queue. (Until this happens, the VME system cpu will not place messages into the queue.)

Queue header

The `IN` pointer (offset from the start of the queue header) points to the next available space in the queue for a message. It is altered by the VME system cpu as the last act upon placing the new message into the queue, after first checking that there is room available to hold the message.

The `OUT` pointer points to the next message to be removed from the queue by the co-processor. It is altered after the co-processor has removed the message from the queue. When the two pointer `IN` and `OUT` are equal, the queue is considered empty. When they are unequal, there is at least one message in the queue.

The LIMIT word is the total size of the queue (in bytes). It is determined by the contents of the VME system table directory.

The START word is the offset to the start of the queue body. When new entries have reached LIMIT, the IN pointer circles back to START.

The KEY word has the value 'MZ' when the queue is initialized, and it is changed by the co-processor to 'MQ' to signal that the queue has been recognized.

The INERR byte counts times when the VME system cpu tried to place an entry into the queue, but found the queue full.

The OUTERR byte is incremented by the co-processor cpu when it encounters an error in processing the messages it removes from the queue.

The INCNT word is incremented for each message successfully placed into the queue.

The OUTCNT word is incremented by the co-processor when it successfully removes a message from the queue.

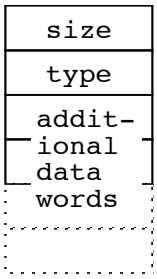
Protocol

When the VME system cpu has a message to place into the queue, it checks to see that the KEY word has the value 'MQ'. It then checks to see if the message can fit. If IN≥OUT, it checks for space between IN and LIMIT; if there is not enough space there, a zero is placed at the word pointed to by IN, and IN is reset to START. If either IN<OUT or IN had to be reset to START, it checks for space between IN and OUT. If there is room, the message is copied into the space, and the IN pointer is advanced by the message size.

The co-processor examines the queue at its convenience. (Note that the co-processor had to have *a priori* knowledge of the location of the queue.) If the queue is not empty (IN≠OUT), a message is removed from the queue. The word pointed to by OUT is examined. If it is zero, OUT is reset to START, and if IN≠OUT, the word at START is examined. When the co-processor has removed the message from the queue, it advances the OUT pointer by the message size. It may then check to see if another message is present. It is assumed that the co-processor will poll the message queue often enough that the queue will not fill up. If it does, one will find the INERR count nonzero.

Message format

Messages placed in the queue for a co-processor conform to a simple structure:



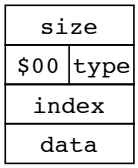
The first word is the message size, including the `size` word. The second word is the message `type`. Additional message contents may follow the second word. So the minimum message size is 4 bytes, in the case that no additional data is required for a given type. The `type` word and any additional data have meaning only to the co-processor, not the VME system.

Generic message setting

To send a general message to a co-processor in a VME system, one has only to send the appropriate setting, specifying listype #40. The ident used with this listype supplies the co-processor number (0,1,2…). The number of data bytes specified in the setting—incremented by 2—becomes the `size` word of the message placed into the selected queue. The first word of the setting data, then, is the `type` word of the message. Additional data words follow the `type` word. Note that the VME system does not care about the `type` word value.

Analog control

There is a format in use for analog control, in which an analog channel may be set which results in a message sent to a co-processor queue.



In this case, the `type` value is given by a byte from the analog control field of the analog descriptor for that channel. The `index` value is given by a word from the analog control field. The data word is the word of setting data in the analog control setting. The `size` value is therefore 8 bytes.